NASA Software Safety Standard

Cynthia Calhoun
NASA Glenn Research Center
Cleveland, Ohio

Fourth Annual Southeastern
Software Engineering Conference
March 29, 2005

Software and System Safety

When a device or system can lead to injury, death, the destruction or loss of vital equipment, or damage to the environment, system safety is paramount.

System safety discipline focuses on "hazards" and the prevention of hazardous situations.

Hardware or software that can lead to a hazard, or is used to control or mitigate a hazard, comes under the "hazard" category.

Software safety discipline expands beyond the immediate software used in hazard control or avoidance to include all software that can impact hazardous software or hardware.

Software must be safe.

"Software does not fail – it just does not perform as intended."
Dr. Nancy Leveson, MIT

Hazardous Software

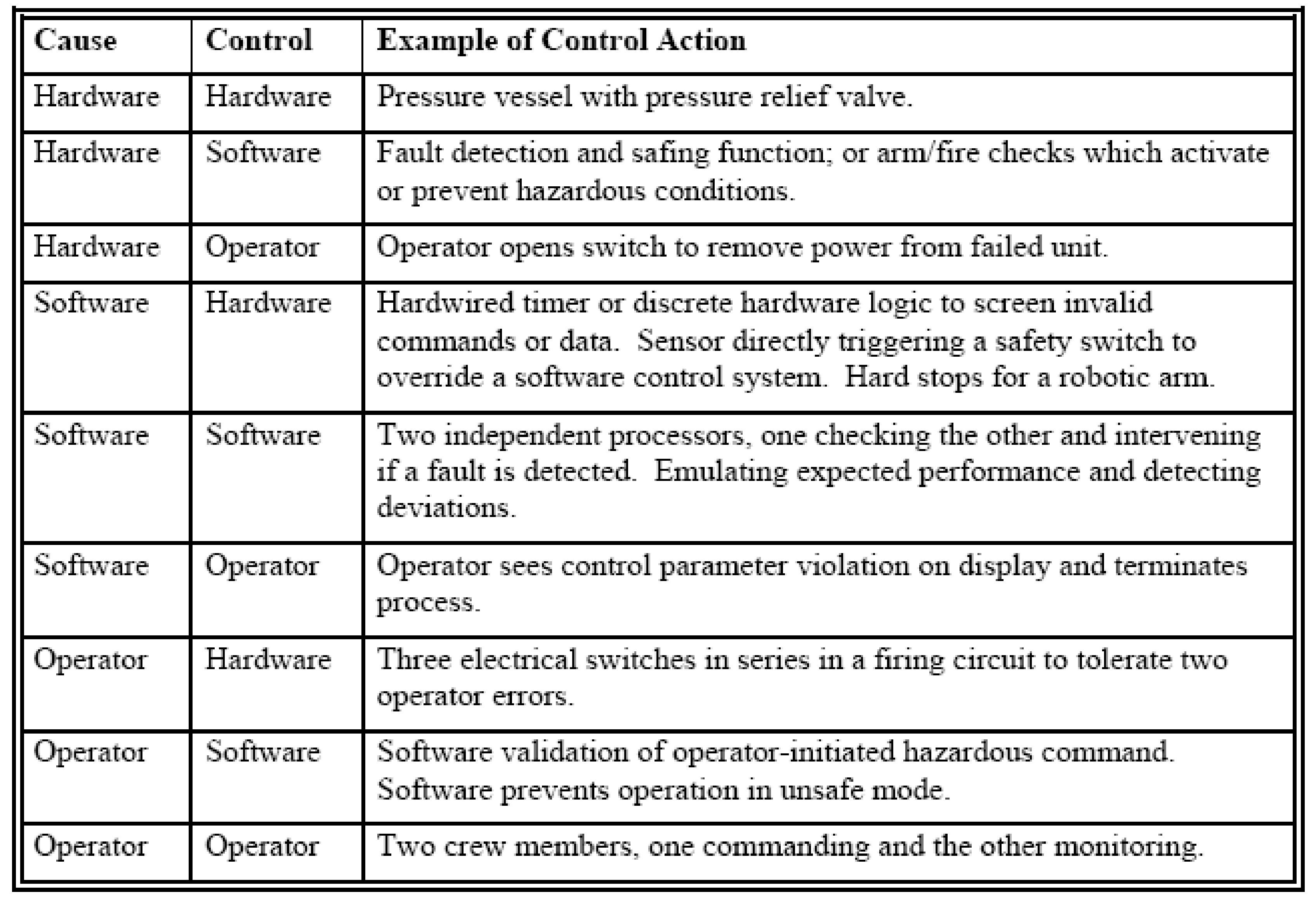
A hazard is the presence of a potential risk situation that can result in or contribute to a mishap. Every hazard has at least one cause, which in turn can lead to a number of effects.

A hazard cause may be a defect in hardware or software, a human operator error, or an unexpected input or event which results in a hazard.

A hazard control is a method for preventing the hazard, reducing the likelihood of the hazard occurring, or the reduction of the impact of that hazard.

Software can be used to detect and control hazards, but software failures can also contribute to the occurrence of hazards.

Hazard Causes and Controls



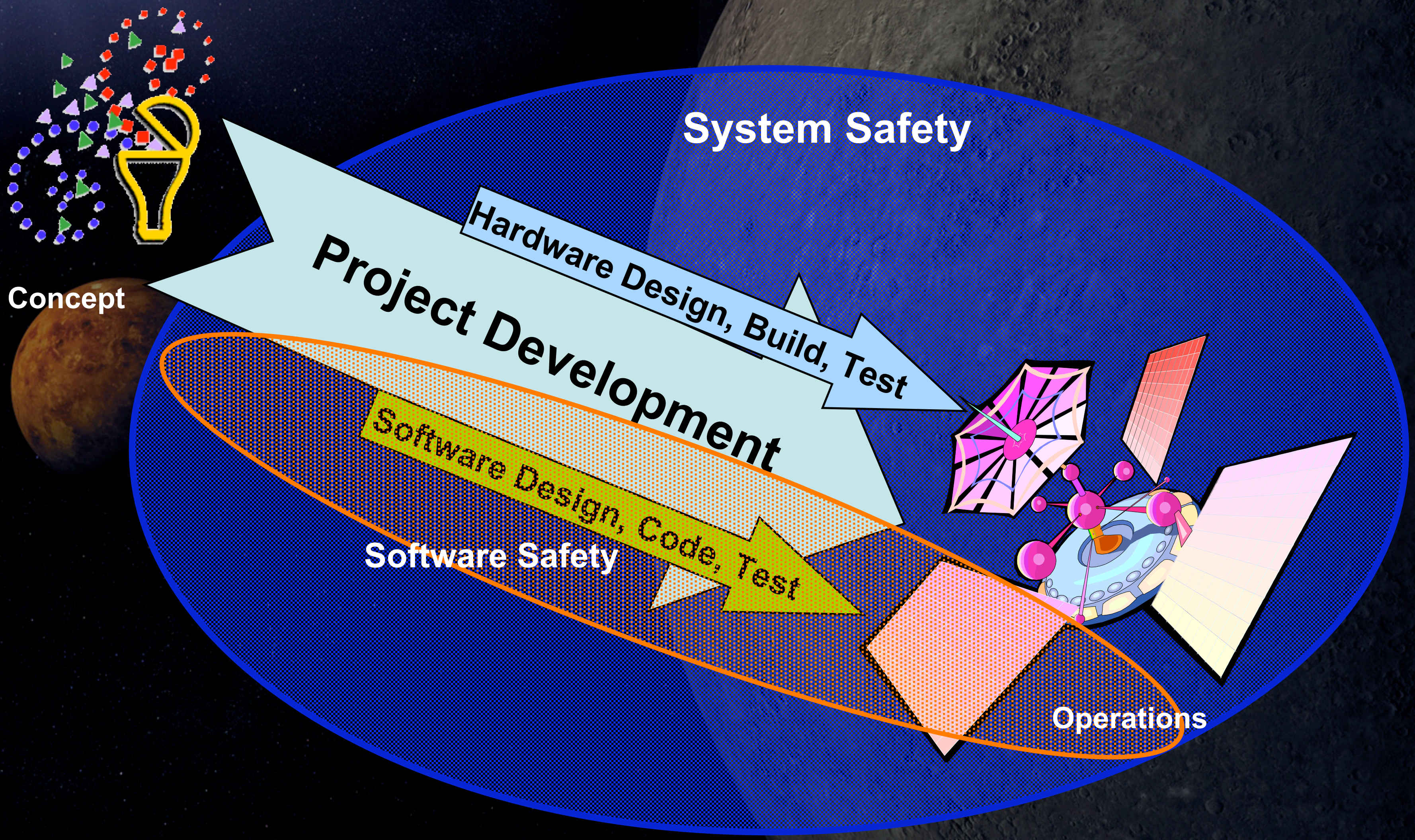
Cause	Control	Example of Control Action
Hardware	Hardware	Pressure vessel with pressure relief valve.
Hardware	Software	Fault detection and safing function; or arm/fire checks which activate or prevent hazardous conditions.
Hardware	Operator	Operator opens switch to remove power from failed unit.
Software	Hardware	Hardwired timer or discrete hardware logic to screen invalid commands or data. Sensor directly triggering a safety switch to override a software control system. Hard stops for a robotic arm.
Software	Software	Two independent processors, one checking the other and intervening if a fault is detected. Emulating expected performance and detecting deviations.
Software	Operator	Operator sees control parameter violation on display and terminates process.
Operator	Hardware	Three electrical switches in series in a firing circuit to tolerate two operator errors.
Operator	Software	Software validation of operator-initiated hazardous command. Software prevents operation in unsafe mode.
Operator	Operator	Two crew members, one commanding and the other monitoring.

Software Safety Discipline

The aspects of software engineering and software assurance that provide a systematic approach to identifying, analyzing, and tracking software mitigation and control of hazards and hazardous functions (e.g., data and commands) to ensure safer software operation within a system.

NASA-STD-8719.13B Software Safety Standard provides requirements that will ensure the safety-critical software receives the required levels of attention throughout the project life cycle.

Software Safety and Project Development



K. Berens – “Enhancing Safety in Software-Intensive Systems”

The

NASA-STD-8719.13B Software Safety Standard

Overview

Developed by the NASA Office of Safety and Mission Assurance to provide the requirements for software safety across all NASA Centers, programs and facilities.

Provides requirements to implement a systematic approach to software safety as an integral part of the project's overall system safety program, software development, and software assurance processes.

Available for Project Managers, Software Engineers, Software Assurance Engineers, System Engineers, and safety practitioners to utilize as requirements for assessing software's contribution to safety and quality.

Purpose

Describes the activities necessary to ensure that safety is designed into the software that is acquired or developed by NASA and that safety is maintained throughout the software and system life cycle.

Specifies the software safety activities, data, and documentation necessary for the acquisition or development of software in a safety-critical system.

Contains two types of safety requirements:

- Process Oriented Requirements - what needs to be done to ensure software safety.
- Technical Requirements - what the system must include or implement (e.g., two-fault tolerance).
- Both need to be addressed and properly documented within a program, project, or facility.

Applicability

Applicable to all safety-critical software acquired or developed by NASA.

- Flight, Ground Support, and Facilities software.
- Software that resides in hardware (i.e., firmware).
- Government furnished software.
- Purchased software (including commercial-off-the-shelf (COTS) software).
- Reused software when included in a safety-critical NASA system.
- Software in a system already in development or is a part of a legacy system.

Tailoring

While the requirements of the NASA Software Safety Standard *can not* be tailored, the activities performed to meet the requirements in the Standard can and should be tailored.

– Although the requirements must be met, the implementation and approach to meeting these requirements may and should vary to reflect the system to which they are applied.

– The level of risk posed by the safety-critical software will be a function of the hazard criticality and the degree of control the software has over the safety functions of the system.

Safety-Critical Software [1]

1. Resides in a safety-critical system (as determined by a hazard analysis) AND at least one of the following:
   a. Causes or contributes to a hazard.
   b. Provides control or mitigation for hazards.
   c. Controls safety-critical functions.
   d. Processes safety-critical commands or data.
   e. Detects and reports, or takes corrective action, if the system reaches a specific hazardous state.
   f. Mitigates damage if a hazard occurs.
   g. Resides on the same system (processor) as safety-critical software.

Safety-Critical Software [2]

2. Processes data or analyzes trends that lead directly to safety decisions (e.g., determining when to turn power off to a wind tunnel to prevent system destruction).

3. Provides full or partial verification or validation of safety-critical systems, including hardware or software subsystems.

Assume the software is safety-critical.
Prove that it is not.

Software Safety Management

Organization and Responsibilities

Software Safety Planning
Personnel Qualifications and Training
Resources
Software Life Cycles
Documentation Requirements
Traceability
Discrepancy and Problem Reporting and Tracking
Software Configuration Management Activities
Software Assurance Activities
Tool Support and Approval
Off-the-shelf Software (COTS/GOTS/OTS)
Contract Management
Certification Process
Waivers/Deviations
Security

Software Development & Safety Analyses

Describes which software safety tasks to perform for each software development life cycle phase, using the waterfall life cycle as the primary life cycle methodology.

- Software Safety Requirements and Analysis
- Software Design and Safety Analysis
- Software Implementation and Safety Analysis
- Software Test and Safety Analysis

Tracing system should be used to:

- Trace the flow down of the software safety requirements to design, implementation, and test.
- Map the relationships between software safety requirements and system hazard reports.

Software Safety Requirements & Analysis

Performed in conjunction with or immediately following the system hazard analysis.

Software safety requirements carry a unique identification in the software requirements specification for traceability purposes.

Software safety requirements should do more than prohibit unsafe behavior.

- Proactively to monitor the system.
- Analyze critical data.
- Look for trends.
- Signal when events occur that may be precursors to a hazardous state.

Software Design and Safety Analysis

Identifies software safety design features and methods.

Software design ensures that software safety features and requirements can be thoroughly tested.

Safety requirements are traced to the design elements that implement the requirement, and each design element is traced back to the requirements which implements it.

Analysis methodology is documented and any improperly designed safety features is documented and reported.

Software Implementation & Safety Analysis

All software safety design features and methods are implemented in the software code.

Software safety personnel are responsible for analyzing the method of implementation, documenting the analysis methodology used, and documenting and reporting any improperly implemented safety features.

Verification of each safety-critical code unit and data is completed prior to the unit's incorporation in a higher-level code package.

Software Test and Safety Analysis

Software testing includes safety testing at the unit level and component level, as well as system and acceptance testing.

All artifacts used to conduct the tests are placed under configuration management.

All functional software safety requirements and safety-critical software elements are verified by testing.

Requirements that cannot be verified by test are verified by evaluation, inspection, or demonstration.

Results from the software and system test process, or the requirements verification evaluation, inspection, or demonstration process, are analyzed.

Operational Use of the Software

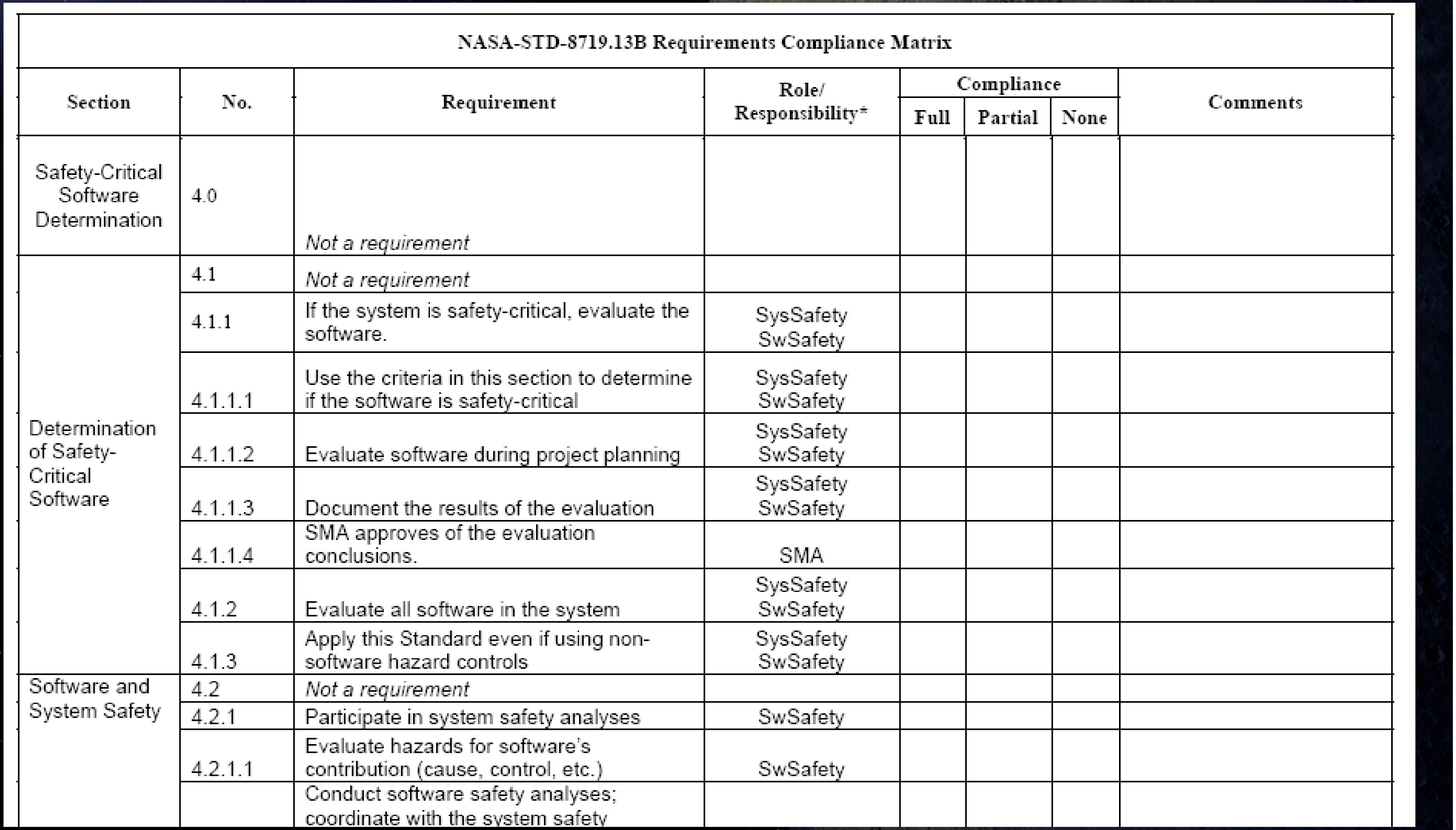
Requirements of this Standard continue to be applicable after the safety-critical software has been released for operations.

Software safety requirements to specify, develop, analyze, and test safety-critical software, applies to all changes made to the software or routine operational updates.

Operational documentation, including user manuals and procedures, describes all safety related commands, data, input sequences, options, and other items necessary for the safe operation of the system.

Requirements of this Standard expire for a particular facility or system only upon the retirement of that facility or system.

Requirements Compliance Matrix



NASA-STD-8719.13B Requirements Compliance Matrix

Section	No.	Requirement	Role/ Responsibility*	Compliance			Comments
				Full	Partial	None	
Safety-Critical Software Determination	4.0	*Not a requirement*					
Determination of Safety-Critical Software	4.1	*Not a requirement*					
	4.1.1	If the system is safety-critical, evaluate the software.	SysSafety SwSafety				
	4.1.1.1	Use the criteria in this section to determine if the software is safety-critical	SysSafety SwSafety				
	4.1.1.2	Evaluate software during project planning	SysSafety SwSafety				
	4.1.1.3	Document the results of the evaluation	SysSafety SwSafety				
	4.1.1.4	SMA approves of the evaluation conclusions.	SMA				
	4.1.2	Evaluate all software in the system	SysSafety SwSafety				
	4.1.3	Apply this Standard even if using non-software hazard controls	SysSafety SwSafety				
Software and System Safety	4.2	*Not a requirement*					
	4.2.1	Participate in system safety analyses	SwSafety				
	4.2.1.1	Evaluate hazards for software's contribution (cause, control, etc.)	SwSafety				
		Conduct software safety analyses; coordinate with the system safety					

Conclusions

Requirements specified in this Standard:

- Ensure that software is considered within the context of system safety, and that appropriate measures are taken to create safe software.
- Ensure that software safety is addressed in project planning, management, and control activities.
- Ensure that software safety is considered throughout the system life cycle, including generation of requirements, design, coding, test, and operation of the software.
- Ensure that software acquisitions, whether off-the-shelf or contracted, have evaluated, assessed, and addressed the software for its safety contributions and limitations.

Location of the Standard

NASA-STD-8719.13B

NASA Software Safety Standard

http://www.hq.nasa.gov/office/codeq/doctree/871913.htm